# Snapshot Handling by Windows Planed Task and PowerShell

The following examples are meant as a functional guide. Do not use the scripts for production use. The scripts are simplified and do not include any error handling.

## API initialization and Authentication

It is possible to write a script using clear text password in the script. I recommend storing the password encrypted in a file (using DPAPI). To do so, you can create the password file (in his example $HOMEPATH\PB\_API.pwd) using:

# get Credentials

$\_creds = Get-Credential -Message "ProfitBricks Account"

# create a secured string from Credentials.Password and store in a File

$\_creds.Password | ConvertFrom-SecureString | Set-Content "$env:HOMEPATH\PB\_API.pwd"

The encryption is bound to the user context. So only the same user can encrypt the data. If you plan to run scrip based on these credentials you have to create the file using the same windows account the script is impersonating later.

To create a Class based on the SOAP WSDL use the following code in your api-script:

# Credentials

$\_password = Get-Content "$env:HOMEPATH\PB\_API.pwd" | ConvertTo-SecureString

$\_user = user@domain.top

$pb\_creds = New-Object System.Management.Automation.PsCredential($\_user,$\_password)

# API WSDL and Endpoint

$pb\_wsdl = "https://api.profitbricks.com/1.3/wsdl"

$PBEndpoint = "https://api.profitbricks.com/1.3/"

# connect Webservice

$pb\_api = New-WebServiceProxy -Uri $pb\_wsdl -namespace ProfitbricksApiService -class ProfitbricksApiServiceClass -Credential $pb\_creds

# configure Webservice

$pb\_api.Url = $PBEndpoint

$pb\_api.PreAuthenticate = $true

$pb\_api.Credentials = $pb\_creds

# $pb\_api is the Webserice class with all methodes and properties defined in the WSDL

## Creating Snapshots

Selecting the storages to snapshot depends on the use case. In the following example the use case is to snapshot all storages in a defined datacenter. To get a list of all your datacenters use

$pb\_api.getAllDataCenters() | Format-Table -Property dataCenterName,dataCenterId

The script does evaluate all available storages in the datacenter ad will run a “createSnapshot” Request for each available storage.

foreach ($storage in $Datacenter.storages) {

# define snapshot parameters

$snapshotRequest = New-Object ProfitBricksApiService.createSnapshotRequest

$snapshotRequest.storageId = $storage.storageId

$snapshotRequest.snapshotName = "Backup." + $storage.storageName +"." + $storage.storageId

$snapshotRequest.description = "Auto created backup snapshot from " + $storage.storageName + " at " + (Get-Date).ToString()

# execute snapshot

$snapshotResponse = $pb\_api.createSnapshot($snapshotRequest)

Write-Host "Created Snapshot from Storage" $storage.storageName "using ID" $storage.storageId "in Snapshot" $snapshotResponse.snapshotId

}

## Delete old Snapshots

You can retrieve a list of all available snapshots using

$pb\_api.getAllSnapshots()

From this list you can select the snapshots to delete. In this example all snapshots with a creation timestamp older than 5 days and a name starting with “Backup.” will be deleted.

# Days for Snapshots to keep

$DaysToArchive = 5

# Snapshot names to work with beginn with string

$SnapshotNames = "Backup.\*"

# timestamp for deletion

$Now = Get-Date

$SnapshotDeleteTime = $now.AddDays($DaysToArchive\*-1)

# get All Snapshots

$AllSnapshots = $pb\_api.getAllSnapshots()

# get Snapshots to delete

$SnapshotsToDelete = $AllSnapshots | Where-Object {$\_.snapshotName -like $SnapshotNames -and $\_.creationTimestamp -lt $SnapshotDeleteTime }

# loop throught all snapshots to delete

foreach ($snapshot in $SnapshotsToDelete) {

$deleteResponse = $pb\_api.deleteSnapshot($snapshots.snapshotId)

Write-Host "Request" $deleteResponse.requestId "Deleted Snapshot" $snapshot.snapshotName "with ID" $snapshot.snapshotId

}

## Run a PowerShell script as planed Task

To run a PowerShell Script as planed Task the Program to start is:

powershell.exe

And the arguments are:

-Noninteractive –Command “&{FullPathToYourScript}”

